Proiect final Probabilități și Statistică

Vultur Sofia-Maria, Trandafir Dragos, Lascu Matei

Grupa 243

1) Tema abordată

Proiectul constă în realizarea unei aplicaţii care să ilustreze modalitatea de generare a unei variabile aleatoare repartizate normal, folosind algoritmul de acceptare-respingere și metoda Box-Muller.

2) Noțiunile din curs și laborator ilustrate în redactarea proiectului

1. Definiția densităţii repartiţiei normale $N(\mu,\sigma^2)$

$$f_X(x) \left(\overset{not}{=} \varphi(x)\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}.$$

2. Funcţia *runif* : folosită pentru generarea de valori aleatoare pe [0,1]
3. Funcţia *hist :* utilizată la generarea unor histograme pe baza unui set de date
4. Funcţia *plot:* integrată pentru realizarea graficelor unor funcţii

3) Algoritmul de acceptare și respingere

Aceasta reprezintă o metodă de bază pentru construirea de variabile aleatoare, bazată pe generarea unor valori aleatoare prin funcţia de repartiţie deja cunoscută a variabilei căutate, îngradite strategic de valorile unei funcţiei de densitate ale alte variabile aleatoare, mai uşor de estimate, înmulţită cu o constantă c alesă astfel : $f(x) \leq cg(x)$.

Prin urmare, pentru generarea unei repartiţii normale, se optează convenabil pentru o variabilă cu distrubuţie exponenţială, de $\Lambda=1$, drept suport, care să mărginească valorile generate aleator.

```r
f <- function(x){ #repartia principala
  sqrt(2/pi) * exp(-x ^ 2 / 2)
}

g <- function(x){ #repartitia suport
  exp(-x)
```

```
}
```

Fie v vectorul de valori prin care se va defini distribuţia normală aproximată prin metoda de acceptare şi respingere:

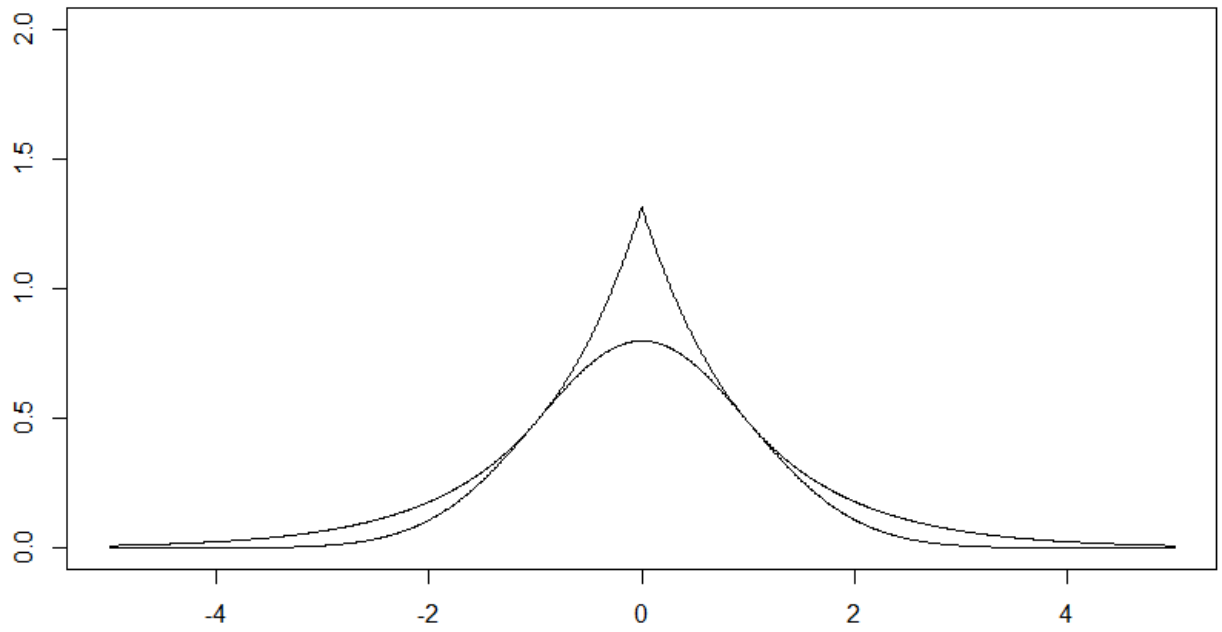
```
v <- rep(0, 100000) #vectorul pentru elementele acceptate
```

Acesta va fi populat după algoritmul enunţat, notabilă fiind determinarea valorii care impune condiţia de acceptare:

```
for (i in 1 : length(v)){
  repeat{ #testarea unui sample
    x <- -log(runif(1))
    y <- runif(1)
    if(y < f(x)/(f(1) / g(1) * g(x))) #conditia de acceptare
    {break;}
  }
  v[i] <- ((-1) ^ (runif(1) < 0.5)) * x
}
```

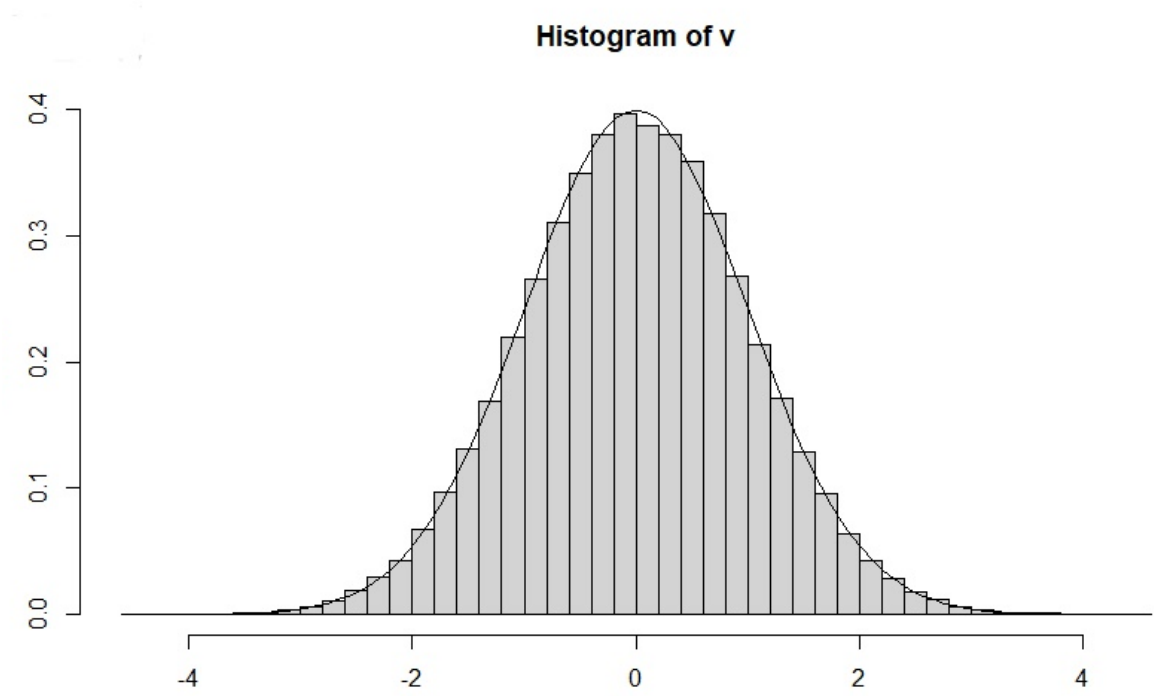
Aproximarea reuşită a normalei este ilustrată atât prin suprapunerea graficelor exponenţialei alese cu funcţia generată de algoritm

```
plot(range1, f(range1), type = 'l',  ylim = c(0, 2))
lines(range2, f(1) / g(1) * g(range2))
lines(-range2, f(1) / g(1)* g(range2))
```



cât şi prin realizarea unei histograme pentru setul de valori ce defineşte distribuţia generată, confruntată cu graficul unei normale realizată cu funcţia predefinită dnorm:

```
hist(v, prob = T, breaks = 50)
curve(dnorm(x,0,1), add = T)
```



4) Algoritmul Box-Muller

Această cea de-a doua metodă vizează exclusiv generarea de distribuţii gaussiene şi constă în alegerea a două variabile U_1, U_2, uniform distrubuite, cu valori cuprinse între (0,1), respectiv construirea unei aplicaţii

$$g : (0, 1) \times (0, 1) \to \mathrm{R}^2$$

astfel încat $(Y_1, Y_2) = g(U_1, U_2)$ sa fie normal distribuite. Cei doi matematicieni aleg bijecţia

$$g(\mathrm{u}, \mathrm{v}) = (\sqrt{-2 \ln \mathrm{u}} \cos(2\pi \mathrm{v}), \sqrt{-2 \ln \mathrm{u}} \sin(2\pi \mathrm{v})).$$

Fie 10000 numărul de valori pentru care urmează a fi generată normala, set de valori reţinut în listele x şi y, iar valorile celor două variabile generate cu funcţia runif.

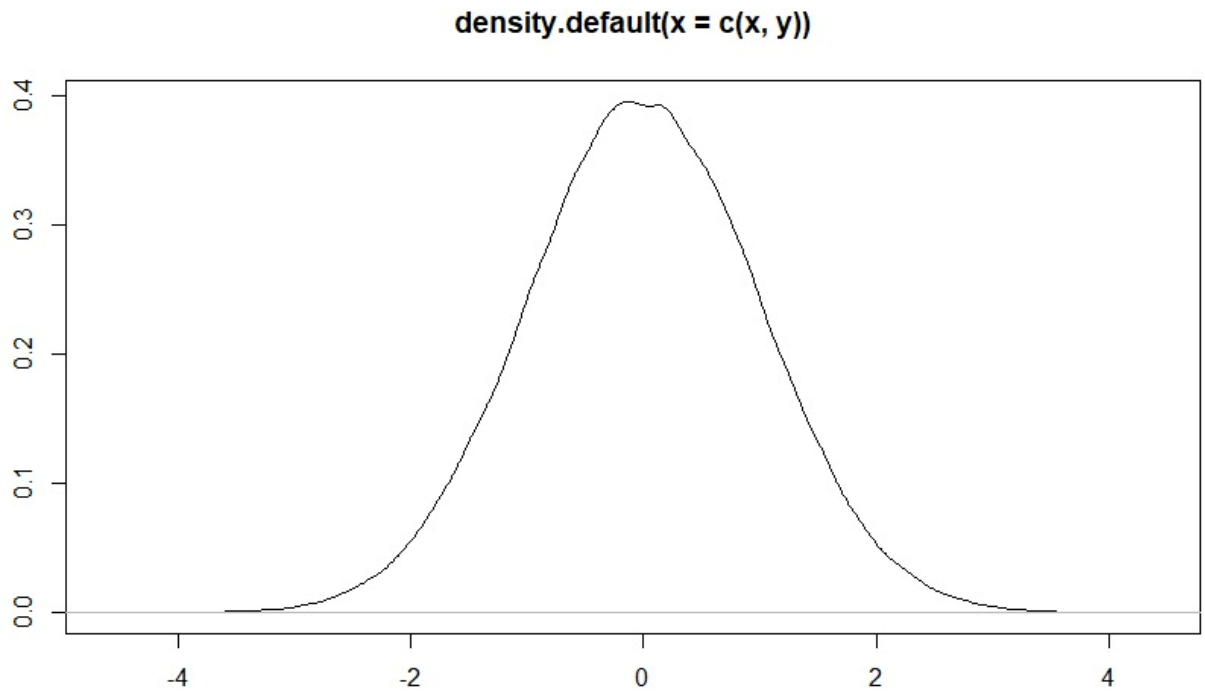
```
range = 100000
u = runif(range)
v = runif(range)
x=rep(0,range)
y=rep(0,range)
```

În continuare, algoritmul atribuie listei x domeniul de valori şi listei y imaginea funcţiei, după definiţia lui g menţionată mai sus.

```
for (i in 1:range){
  x[i] = sqrt(-2*log(u[i]))*cos(2*pi*v[i])
  y[i] = sqrt(-2*log(u[i]))*sin(2*pi*v[i])
}
```

Rezultatul este ilustrat prin plotarea unei funcţii care are pentru x[i] returnează valoarea lui y[i].

```
plot(density(c(x,y)))
```



5) Bibliografie

- https://rmarkdown.rstudio.com/authoring_quick_tour.html
- https://www.middleprofessor.com/files/applied-biostatistics_bookdown/_book/getting-started-r-projects-and-r-markdown.html

- https://www.math.arizona.edu/~tgk/mc/book_chap4.pdf
- https://en.wikipedia.org/wiki/Rejection_sampling#Algorithm